Database Systems

Lecture #3

Dr Sherin ElGokhy

Chapter 3: Relational Algebra and Calculus

Query languages for relational databases

- Operations on databases:
  - queries: "read" data from the database
  - updates: change the content of the database
- An update can be seen as a functions that, given a database produce another database (without changing the schema).
- A query can be seen as a functions that, given a database produce a relation.

Query languages for relational databases

- Foundations can be studied with reference to query languages:
  - **Relational Algebra**, a "procedural" language, in which the data retrieval functions are specified by describing the procedure that must be followed in order to obtain the result.
  - **Relational Calculus**, a "declarative" language, in which the data retrieval functions describe the properties of the results, rather than the procedure used to obtain it.
  - (briefly) **Datalog**, a more powerful language, which allows the formulation of queries that could not be expressed in algebra or in calculus.
- Then, we will study **SQL**, a practical language (with declarative and procedural features) for queries and updates

Relational Algebra

- Relational Algebra is a procedural language, based on algebraic concepts.
- Consists of a collection of operators that
  - are defined on relations
  - produce relations as results and therefore expressions can be combined to form complex expressions
- Operators
  - **union, intersection, difference**
  - **renaming**
  - **selection**
  - **projection**
  - **join (Natural join, Cartesian product, Theta join,…)**

Union, intersection, difference

- **Relations are sets**, so we can apply set operators
- However, we want the results to be relations (that is, homogeneous sets of tuples)
- Therefore:
  - it is meaningful to apply union, intersection, difference only to pairs of relations defined over the same attributes (homogeneous sets of tuples).

Union

GRADUATES

Number	Surname	Age
7274	Raafat	37
7432	Hegazy	39
9824	Shawky	38

MANAGERS

Number	Surname	Age
9297	Hegazy	56
7432	Hegazy	39
9824	Shawky	38

GRADUATES ∪ MANAGERS

Number	Surname	Age
7274	Raafat	37
7432	Hegazy	39
9824	Shawky	38
9297	Hegazy	56

Intersection

GRADUATES

Number	Surname	Age
7274	Raafat	37
7432	Hegazy	39
9824	Shawky	38

MANAGERS

Number	Surname	Age
9297	Hegazy	56
7432	Hegazy	39
9824	Shawky	38

GRADUATES ∩ MANAGERS

Number	Surname	Age
7432	Hegazy	39
9824	Shawky	38

Difference

GRADUATES

Number	Surname	Age
7274	Raafat	37
7432	Hegazy	39
9824	Shawky	38

MANAGERS

Number	Surname	Age
9297	Hegazy	56
7432	Hegazy	39
9824	Shawky	38

GRADUATES - MANAGERS

Number	Surname	Age
7274	Raafat	37

A meaningful but impossible union

PATERNITY

Father	Child
Sherif	Tamer
Sherif	Bahaa
Raouf	Maged
Raouf	Omar

MATERNITY

Mother	Child
Faten	Tamer
Faten	Bahaa
Laila	Maged
Nadia	Omar

Paternity ∪ Maternity **???**

- It would be meaningful to execute a sort of union in order to obtain all the 'parent–child' pairs held in the database, but that is not possible.
- The problem: father and mother are different names, but both represent a "parent"
- The solution: rename attributes

Renaming

- unary operator (has one relation as argument).
- "changes attribute names" without changing values
- removes the limitations associated with set operators
- notation:

  $\rho_{Y \leftarrow X}(r)$
- **Example:**

  $\rho_{Parent \leftarrow Father}(Paternity)$
- if there are two or more attributes involved then ordering is meaningful:
  - $\rho_{Location,\ Pay \leftarrow Branch,\ Salary}(Employees)$

Renaming, example

PATERNITY

Father	Child
Sherif	Tamer
Sherif	Bahaa
Raouf	Maged
Raouf	Omar

$\rho_{Parent \leftarrow Father}(PATERNITY)$

Parent	Child
Sherif	Tamer
Sherif	Bahaa
Raouf	Maged
Raouf	Omar

Renaming and union

PATERNITY

Father	Child
Sherif	Tamer
Sherif	Bahaa
Raouf	Maged
Raouf	Omar

MATERNITY

Mother	Child
Faten	Tamer
Faten	Bahaa
Laila	Maged
Nadia	Omar

$$\rho_{Parent \leftarrow Father}(PATERNITY) \cup \rho_{Parent \leftarrow Mother}(MATERNITY)$$

Parent	Child
Sherif	Tamer
Sherif	Bahaa
Raouf	Maged
Raouf	Omar
Faten	Tamer
Faten	Bahaa
Laila	Maged
Nadia	Omar

Renaming and union, with more attributes

EMPLOYEES

Surname	Branch	Salary
Nassar	Cairo	4500
Safwat	London	5300

STAFF

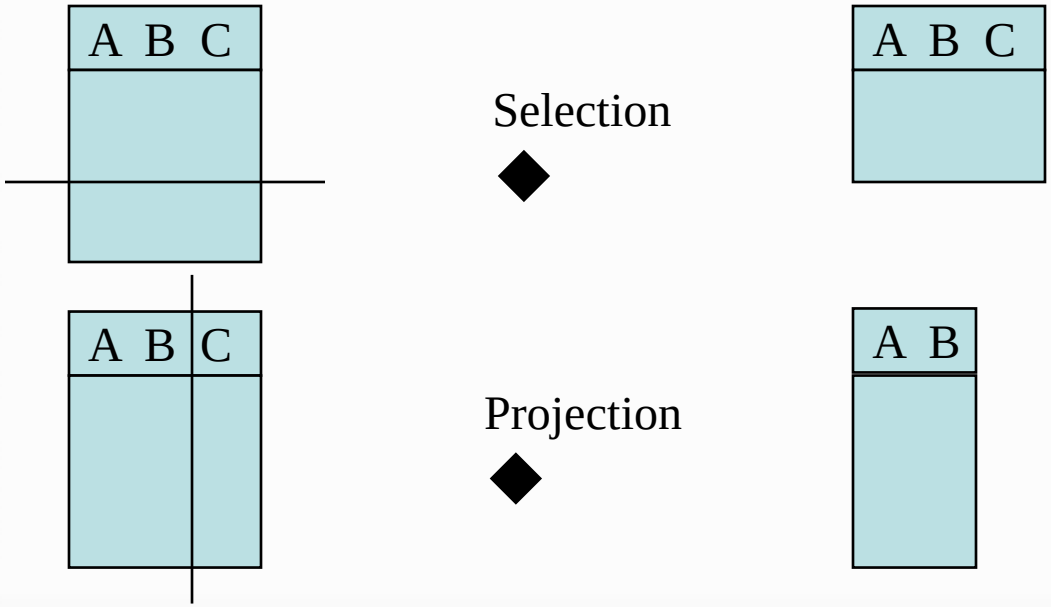
Surname	Factory	Wages
Hamed	Mansoura	3300
Ramzy	Port-Said	3200

$\rho_{\text{Location,Pay} \leftarrow \text{Branch,Salary}}(\text{EMPLOYEES}) \cup \rho_{\text{Location,Pay} \leftarrow \text{Factory, Wages}}(\text{STAFF})$

Surname	Location	Pay
Nassar	Cairo	4500
Safwat	London	5300
Hamed	Mansoura	3300
Ramzy	Port-Said	3200

Selection and projection

- Two unary operators, in a sense orthogonal:
  - selection for "horizontal" decompositions, produces a subset of tuples on all the attributes.
  - projection for "vertical" decompositions , produces a result to which all the tuples contribute, but on a subset of the attributes.



Selection

- Produce results

  a subset of the tuples (those that satisfy a condition) on all the attributes (the same schema as the operand).

- Notation:

  $\sigma_F(r)$

- Semantics:
  - $\sigma_F(r) = \{ t \mid t \in r$ and t satisfies F$\}$
  - F is a formula like: $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)

Selection, example

EMPLOYEES

Surname	FirstName	Age	Salary
Fawzy	Monir	25	2000
Kamel	Sayed	40	3000
Salim	Nabil	36	4500
Abdel-Aziz	Hassan	40	3900

$\sigma_{Age<30 \vee Salary>4000}$ **(EMPLOYEES)**

Surname	FirstName	Age	Salary
Fawzy	Monir	25	2000
Salim	Nabil	36	4500

Selection, another example

CITIZENS

Surname	FirstName	PlaceOfBirth	Residence
Fawzy	Monir	Cairo	Ismailia
Kamel	Sayed	Cairo	Cairo
Selim	Nabil	Alex	Alex
Abdel-Aziz	Hasan	Suez	Alex

$\sigma_{\text{PlaceOfBirth=Residence}}$ **(CITIZENS)**

Surname	FirstName	PlaceOfBirth	Residence
Kamel	Sayed	Cairo	Cairo
Selim	Nabil	Alex	Alex

Selection, Precise definition

Given a relation , a *propositional formula F on X is a* formula obtained by combining atomic conditions of the type *AϑB or Aϑc* with the connectives ∧ (**and**), ∨ (**or**), ¬ (**not**) where:

• ϑ is a comparison operator (=, ≠, >, <, ≤, ≥);

• A and B are attributes in X that are compatible (that is, the comparison ϑ is meaningful on the values of their domains);

• c is a constant compatible with the domain of A.

Given a formula *F and a tuple t, a truth value is defined for F on t:* AϑB is true on t if and only if t[A] is in relation ϑ with t[B](for example, A=B is true on t if and only if t[A] = t[B]);

• A ϑ c is true on t if and only if t[A] is in relation ϑ with c;

• F1 ∨ F2, F1 ∧ F2 and ¬ F1 have the usual meaning.

• the selection $\sigma_F(r)$ produces a relation on the same attributes as r that contains the tuples of r for which F is true.

Projection

- Produce results
  - all the tuples contribute,
  - but on a subset of the attributes.
- **Notation** (given a relation r (X) and a subset of attributes Y of X):

  $\pi_Y(r)$
- **Semantics:**

  $\mathbf{\pi_Y(r) = \{ t[Y] \mid t \in r \}}$

Projection, example

EMPLOYEES

Surname	FirstName	Department	Head
Fawzy	Monir	Sales	Khamis
Kamel	Sayed	Sales	Khamis
Selim	Nabil	Personnel	Marzouk
Abdel-Aziz	Hasan	Personnel	Marzouk

$\pi_{\text{Surname, FirstName}}$**(EMPLOYEES)**

Surname	FirstName
Fawzy	Monir
Kamel	Sayed
Selim	Nabil
Abdel-Aziz	Hasan

Projection, another example

EMPLOYEES

Surname	FirstName	Department	Head
Fawzy	Monir	Sales	Khamis
Kamel	Sayed	Sales	Khamis
Selim	Nabil	Personnel	Marzouk
Abdel-Aziz	Hasan	Personnel	Marzouk

$\pi_{\text{Department, Head}}$ **(EMPLOYEES)**

Department	Head
Sales	Khamis
Personnel	Marzouk

Equal attributes collapse into a single tuple

The result contains fewer tuples than the operand, because all the tuples in the operand that have equal values on all the attributes of the projection give the same contribution to the projection itself.

Cardinality of projections

- The result of a projections contains at most as many tuples as the operand
- It can contain fewer, if several tuples "**collapse**"
- $\pi_Y(r)$ contains as many tuples as r if and only if Y is a superkey for r;
  - this holds also if Y is "by chance" (not defined as a superkey in the schema, but superkey for the specific instance), see the example

Tuples that collapse

STUDENTS

RegNum	Surname	FirstName	BirthDate	DegreeProg
284328	Abdel-Aziz	Fatehy	29/04/1989	Computing
296328	Abdel-Aziz	Rashed	29/04/1989	Computing
587614	Abdel-Aziz	Sayed	01/05/1999	Engineering
934856	Kamel	Sayed	01/05/1999	Fine Art
965536	Kamel	Sayed	05/03/1988	Fine Art

$\pi_{\text{Surname, DegreeProg}}$ **(STUDENTS)**

Surname	DegreeProg
Abdel-Aziz	Computing
Abdel-Aziz	Engineering
Kamel	Fine Art

Tuples that do not collapse, "by chance"

STUDENTS

RegNum	Surname	FirstName	BirthDate	DegreeProg
296328	Abdel-Aziz	Rashed	29/04/1989	Computing
587614	Abdel-Aziz	Sayed	01/05/1999	Engineering
934856	Kamel	Sayed	01/05/1999	Fine Art
965536	Kamel	Sayed	05/03/1988	Engineering

$\pi_{\text{Surname, DegreeProg}}$ **(STUDENTS)**

Surname	DegreeProg
Abde-Aziz	Computing
Abde-Aziz	Engineering
Kamel	Fine Art
Kamel	Engineering

Relational Algebra Review

Basic operations:

- *Renaming* (ρ) : rename the attributes
- *Selection* (**σ**) : gives a subset of rows.
- *Projection* (**π**) : deletes unwanted columns.
- *Difference* (**—**) : tuples in relation 1, but not 2
- *Union* (∪) : tuples in relation 1 or 2 or both.
- *Intersection* (∩) : tuples in both relations.

Additional operations:

- *Cross-product* (×) : combine two relations.
- *Join* (⋈) : like × but only keep tuples where common fields are equal.

Join

- The most typical operator in relational algebra
- allows to establish connections among data in different relations, taking into advantage the "value-based" nature of the relational model
- **Two main versions of the join:**
  - " **Natural**" join: takes attribute names into account
  - " **Theta**" join
- They are all denoted by the symbol $\bowtie$

A natural join

• The tuples in the **natural join** are obtained by combining the tuples with equal values on the common attributes.

R_1

Employee	Department
Fawzy	Sales
Kamel	Production
Badr	Production

R_2

Department	Head
Production	Ihab
Sales	Radwan

$R_1 \bowtie R_2$

Employee	Department	Head
Fawzy	Sales	Radwan
Kamel	Production	Ihab
Badr	Production	Ihab

Natural join: definition

- $r_1(X_1)$, $r_2(X_2)$
- $r_1 \bowtie r_2$ (**natural** join of r_1 and r_2) is a relation defined on X_1X_2 (the union of the two sets):

$$\{ t \text{ on } X_1X_2 \mid t[X_1] \in r_1 \text{ and } t[X_2] \in r_2 \}$$

or, equivalently

$$\{ t \text{ on } X_1X_2 \mid \text{exist } t_1 \in r_1 \text{ and } t_2 \in r_2 \text{ with } t[X_1] = t_1 \text{ and } t[X_2] = t_2 \}$$

Natural join: comments

- The tuples in the result are obtained by **combining tuples** in the operands with ***equal values on the common attributes.***
- The degree of the result of a join is less than or equal to the sum of the degrees of the two operands, because the common attributes of the operands appear only once in the result.
- The common attributes often form a key of one of the operands; **there is a referential constraint between the common attributes**. (remember: references are realized by means of keys, and we join in order to follow references).

Natural join: example

OFFENCES

Code	Date	Officer	Department	Registration
143256	25/10/92	567	75	5694 FR
987554	26/10/92	456	75	5694 FR
987557	26/10/92	456	75	6544 XY
630876	15/10/92	456	47	6544 XY
539856	12/10/92	567	47	6544 XY

CARS

Registration	Department	Owner	Address
6544 XY	75	Cordon Edouard	Rue du Pont
7122 HT	75	Cordon Edouard	Rue du Pont
5694 FR	75	Latour Hortense	Avenue Foch
6544 XY	47	Mimault Bernard	Avenue FDR

OFFENCES ⋈ CARS

Code	Date	Officer	Depart-ment	Regist-ration	Owner	Address
143256	25/10/92	567	75	5694 FR	Latour Hortense	Avenue Foch
987554	26/10/92	456	75	5694 FR	Latour Hortense	Avenue Foch
987557	26/10/92	456	75	6544 XY	Cordon Edouard	Rue du Pont
630876	15/10/92	456	47	6544 XY	Mimault Bernard	Avenue FDR
539856	12/10/92	567	47	6544 XY	Mimault Bernard	Avenue FDR

Natural join: Another example

PATERNITY

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

MATERNITY

Mother	Child
Eve	Cain
Eve	Seth
Sarah	Isaac
Hagar	Ishmael

PATERNITY ⋈ MATERNITY

Father	Child	Mother
Adam	Cain	Eve
Abraham	Isaac	Sarah
Abraham	Ishmael	Hagar

Joins can be "incomplete"

- If a tuple does not have a "counterpart /equivalent" in the other relation, then it does not contribute to the join ("dangling" tuple)

R_1

Employee	Department
Fawzy	Sales
Kamel	Production
Badr	Production

R_2

Department	Head
Production	Ayman
Purchasing	Othman

$R_1 \bowtie R_2$

Employee	Department	Head
Kamel	Production	Ayman
Badr	Production	Ayman

Joins can be **empty**

- As an extreme, we might have that no tuple has a counterpart, and all tuples are ***dangling***

R_1

Employee	Department
Fawzy	Sales
Kamel	Production
Badr	Production

R_2

Department	Head
Marketing	Ayman
Purchasing	Othman

$R_1 \bowtie R_2$

Employee	Department	Head

The other extreme

- If *each tuple of each operand can be combined with all the tuples of the other*, then the join has a cardinality that is the **product** of the cardinalities of the operands

R_1

Employee	Department
Fawzy	A
Kamel	A
Badr	A

R_2

Department	Head
A	Ayman
A	Othman

$R_1 \bowtie R_2$

Employee	Department	Head
Fawzy	A	Ayman
Kamel	A	Ayman
Badr	A	Ayman
Fawzy	A	Othman
Kamel	A	Othman
Badr	A	Othman

How many tuples in a join?

Given $r_1(X_1)$, $r_2(X_2)$

- the join has a cardinality between zero and the products of the cardinalities of the operands:

$$0 \leq |r_1 \bowtie r_2| \leq |r_1| \times |r_2|$$

(| r | is the cardinality of relation r)

- moreover:
  - if the join is complete, then its cardinality is at least the maximum of $|r_1|$ and $|r_2|$
  - if $X_1 \cap X_2$ contains a key for r_2, then $|r_1 \bowtie r_2| \leq |r_1|$
  - if $X_1 \cap X_2$ is the primary key for r_2, and there is a referential constraint between $X_1 \cap X_2$ in r_1 and such a key, then $|r_1 \bowtie r_2| = |r_1|$

Outer joins

- A variant of the join, to keep all pieces of information from the operands.
- This allows for the possibility that all the tuples contribute to the result, extended with null values where there is no counterpart.
- It "pads with nulls" the tuples that have no counterpart
- **Three variants:**
  - "**Left**": only tuples of the first operand are padded
  - "**Right**": only tuples of the second operand are padded
  - "**Full**": tuples of both operands are padded

Outer joins

R_1

Employee	Department
Fawzy	Sales
Kamel	Production
Badr	Production

R_2

Department	Head
Production	Ayman
Purchasing	Othman

$R_1 \bowtie_{\mathbf{LEFT}} R_2$

Employee	Department	Head
Fawzy	Sales	**NULL**
Kamel	Production	Ayman
Badr	Production	Ayman

$R_1 \bowtie_{\mathbf{RIGHT}} R_2$

Employee	Department	Head
Kamel	Production	Ayman
Badr	Production	Ayman
NULL	Purchasing	Othman

$R_1 \bowtie_{\mathbf{FULL}} R_2$

Employee	Department	Head
Fawzy	Sales	**NULL**
Kamel	Production	Ayman
Badr	Production	Ayman
NULL	Purchasing	Othman

N-ary join

- The natural join is
  - **Commutative**: $R_1 \bowtie R_2 = R_2 \bowtie R_1$
  - **Associative**: $(R_1 \bowtie R_2) \bowtie R_3 = R_1 \bowtie (R_2 \bowtie R_3)$
- Therefore, we can write n-ary joins without ambiguity:

$$R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n$$

N-ary join

R_1

Employee	Department
Fawzy	Sales
Kamel	Production
Ahmed	Marketing
Badr	Production

R_2

Department	Division
Production	A
Marketing	B
Purchasing	B

R_3

Division	Head
A	Ayman
B	Othman

$R_1 \bowtie R_2 \bowtie R_3$

Employee	Department	Division	Head
Kamel	Production	A	Ayman
Ahmed	Marketing	B	Othman
Badr	Production	A	Ayman

Cartesian product

- The **Natural join** is defined also when the operands have no attributes in common

- in this case no condition is imposed on tuples, and therefore the result contains tuples obtained by combining the tuples of the operands in all possible ways

Cartesian product: Example

EMPLOYEES

Employee	Project
Fawzy	A
Ahmed	A
Ahmed	B

PROJECTS

Code	Name
A	Adel
B	Mahmoud

EMPLOYES × PROJECTS

Employee	Project	Code	Name
Fawzy	A	A	Adel
Ahmed	A	A	Adel
Ahmed	B	A	Adel
Fawzy	A	B	Mahmoud
Ahmed	A	B	Mahmoud
Ahmed	B	B	Mahmoud

Theta-join

- In most cases, a Cartesian product is meaningful only if followed by a selection: which preserves only the combined tuples that satisfy the given requirements.
- For example, it makes sense to define a Cartesian product on the relations EMPLOYEES and PROJECTS, if it is followed by the selection that retains only the tuples with equal values on the attributes Project and Code.
- For this reason, another operator is often introduced, the ***theta-join.***

Theta-join

– **Theta-join:** a derived operator, defined by means of other operators, it is a Cartesian product followed by a selection.

$$R_1 \bowtie_F R_2 = \sigma_F(R_1 \bowtie R_2)$$

➢ A theta-join in which the condition of selection *F is a conjunction of atoms* of equality, each with an attribute of the first relation and one of the second, is called *an* **Equi-join.**

Equi-join: example

EMPLOYEES

Employee	Project
Fawzy	A
Ahmed	A
Ahmed	B

PROJECTS

Code	Name
A	Adel
B	Mahmoud

EMPLOYES $\bowtie_{\text{Project=Code}}$ PROJECTS

Employee	Project	Code	Name
Fawzy	A	A	Adel
Ahmed	A	A	Adel
Ahmed	B	B	Mahmoud

Thanks